```
UUU          UUU  VVV         VVV       111       RRRRRRRRRRR       OOOOOOOOO     MMM         MMM
UUU          UUU  VVV         VVV       111       RRRRRRRRRRRR      OOOOOOOOO     MMM         MMM
UUU          UUU  VVV         VVV       111       RRRRRRRRRRRR      OOOOOOOOO     MMM         MMM
UUU          UUU  VVV         VVV    111111       RRR         RRR   OOO       OOO   MMMMMMM   MMMMMMM
UUU          UUU  VVV         VVV    111111       RRR         RRR   OOO       OOO   MMMMMMM   MMMMMMM
UUU          UUU  VVV         VVV    111111       RRR         RRR   OOO       OOO   MMMMMMM   MMMMMMM
UUU          UUU  VVV         VVV       111       RRR         RRR   OOO       OOO   MMM MMM MMM   MMM
UUU          UUU  VVV         VVV       111       RRR         RRR   OOO       OOO   MMM  MMM  MMM   MMM
UUU          UUU  VVV         VVV       111       RRR         RRR   OOO       OOO   MMM   M   MMM   MMM
UUU          UUU  VVV         VVV       111       RRRRRRRRRRRR      OOO       OOO   MMM       MMM   MMM
UUU          UUU  VVV         VVV       111       RRRRRRRRRRRR      OOO       OOO   MMM       MMM   MMM
UUU          UUU  VVV         VVV       111       RRRRRRRRRRRR      OOO       OOO   MMM       MMM   MMM
UUU          UUU  VVV         VVV       111       RRR   RRR         OOO       OOO   MMM       MMM   MMM
UUU          UUU  VVV         VVV       111       RRR   RRR         OOO       OOO   MMM       MMM   MMM
UUU          UUU  VVV         VVV       111       RRR   RRR         OOO       OOO   MMM       MMM   MMM
UUU          UUU     VVV   VVV          111       RRR      RRR      OOO       OOO   MMM       MMM   MMM
UUU          UUU     VVV   VVV          111       RRR      RRR      OOO       OOO   MMM       MMM   MMM
UUU          UUU     VVV   VVV          111       RRR      RRR      OOO       OOO   MMM       MMM   MMM
UUUUUUUUUUUUUUUU        VVV            111111111   RRR         RRR      OOOOOOOOO     MMM       MMM   MMM
UUUUUUUUUUUUUUUU        VVV            111111111   RRR         RRR      OOOOOOOOO     MMM       MMM   MMM
UUUUUUUUUUUUUUUU        VVV            111111111   RRR         RRR      OOOOOOOOO     MMM       MMM   MMM
UUUUUUUUUUUUUUUU        VVV            111111111   RRR         RRR      OOOOOOOOO     MMM       MMM   MMM
```

```
  CCCCCCCC    OOOOOO    NN      NN    IIIIII      OOOOOO
  CCCCCCCC    OOOOOO    NN      NN    IIIIII      OOOOOO
CC           OO      OO  NN      NN      II      OO      OO
CC           OO      OO  NN      NN      II      OO      OO
CC           OO      OO  NNNN    NN      II      OO      OO
CC           OO      OO  NNNN    NN      II      OO      OO
CC           OO      OO  NN  NN  NN      II      OO      OO
CC           OO      OO  NN  NN  NN      II      OO      OO
CC           OO      OO  NN    NNNN      II      OO      OO
CC           OO      OO  NN    NNNN      II      OO      OO
CC           OO      OO  NN      NN      II      OO      OO      ....
  CCCCCCCC    OOOOOO    NN      NN    IIIIII      OOOOOO        ....
  CCCCCCCC    OOOOOO    NN      NN    IIIIII      OOOOOO        ....

LL             IIIIII      SSSSSSSS
LL             IIIIII      SSSSSSSS
LL               II      SS
LL               II      SS
LL               II      SS
LL               II        SSSSSS
LL               II        SSSSSS
LL               II              SS
LL               II              SS
LL               II              SS
LL               II              SS
LLLLLLLLLL     IIIIII    SSSSSSSS
LLLLLLLLLL     IIIIII    SSSSSSSS
```

Table of contents

```
00000001  0000     1          BOOT_UV1_SWITCH = 1             ; Build Micro VAX I bootstrap emulator
00000001  0000     2          PQ              == 1
                   1          .title  CONIO - console input output routines
                   2          .ident  /V1.0-00/
                   3
                   4 ;
                   5 ;*****************************************************************************
                   6 ;*                                                                           *
                   7 ;*  Copyright (C) 1978, 1980, 1982, 1984                                     *
                   8 ;*                                                                           *
                   9 ;*  Digital Equipment Corporation, Maynard, Massachusetts.                   *
                  10 ;*  all rights reserved.                                                     *
                  11 ;*                                                                           *
                  12 ;*  This software is furnished under a license and may be used and copied    *
                  13 ;*  only in  accordance with  the  terms  of  such  license  and with the    *
                  14 ;*  inclusion of the above copyright notice. This software or  any  other    *
                  15 ;*  copies thereof may not be provided or otherwise made available to any    *
                  16 ;*  other person.  No title to and ownership of  the  software is  hereby    *
                  17 ;*  transferred.                                                             *
                  18 ;*                                                                           *
                  19 ;*  The information in this software is  subject to change without notice    *
                  20 ;*  and  should  not  be  construed as  a commitment by Digital Equipment    *
                  21 ;*  Corporation.                                                             *
                  22 ;*                                                                           *
                  23 ;*  Digital assumes no responsibility for the use  or  reliability of its    *
                  24 ;*  software on equipment which is not supplied by Digital.                  *
                  25 ;*                                                                           *
                  26 ;*****************************************************************************
                  27 ;
                  28 ; Facility:  system bootstrapping
                  29 ;
                  30 ; Abstract: CONIO provides basic console read, readprompt and write facilities.
                  31 ;
                  32 ; Author:  Richard I. Hustvedt, creation date:  27-apr-1978
                  33 ;
                  34 ; Modified by:
                  35 ;
                  36 ;       David N. Cutler 29-Dec-83
                  37 ;
                  38 ;               Add support for QVSS as the console terminal on MicroVax I.
                  39 ;
                  40 ; Include files:
                  41 ;
                  42
                  43          $prdef                          ; define processor registers
                  44          $ssdef                          ; define status code values
                  45
                  46 ;
                  47 ; Equated symbols:
                  48 ;
                  49
0000000D          50          cr      = 13                    ; character code for carriage return
0000000A          51          lf      = 10                    ; character code for line feed
00000015          52          control_u = 21                  ; character code for control-u
00000013          53          control_s = 19                  ; control s (xoff)
00000011          54          control_q = 17                  ; control q (xon)
0000007F          55          rubout = 127                    ; character code for rubout
```

```
00000000  0000    56          v_rub  = 0                    ; rubout sequence in progress
```

```
                                        .sbttl  boo$readprompt - prompt and read input string
                                ;+
                                ;       boo$readprompt outputs the specified asciz prompt string on the
                                ;       console terminal then checks the count of characters to be read.
                                ;       If zero it exits, otherwise it reads the console terminal until
                                ;       either a carriage return is encountered or the character count
                                ;       is satisfied.  The specified buffer is filled with an ascic
                                ;       string containing the characters read but not including the
                                ;       terminating carriage return.
                                ;
                                ; Calling sequence:
                                ;
                                ;       callx   arglist,boo$readprompt
                                ;
                                ; Input parameters:
                                ;
                                ;       prompt(ap)  -  address of asciz prompt string
                        00000004        prompt  =  4
                                ;
                                ;       size(ap)    -  maximum length of input string
                        00000008        size    =  8
                                ;                       note: if size is zero, then nothing is read
                                ;                             and only the prompt string is written.
                                ;
                                ;       buf(ap)     -  address of input buffer
                        0000000C        buf     =  12
                                ;
                                ;       option(ap)  - processor switch value.
                        00000010        option  =  16
                                ;
                                ; Output parameters:
                                ;
                                ;       r0 - completion status code (always ss$_normal)
                                ;
                                ;       Buffer located by buf(ap) will be filled with the string
                                ;       read as an ascic string.
                                ;
                                
                                        .psect  $conio,byte
                        00000000
                        0314    0000    .entry  boo$readprompt,^m<r2,r4,r8,r9>
58      04 AC   D0      0002    10$:    movl    prompt(ap),r8           ;get prompt string address
                54      D4      0006            clrl    r4              ;clear control flags
        50      88      9A      0008    20$:    movzbl  (r8)+,r0        ;get next output character
                05      13      000B            beql    30$             ;if eql none
                0086    30      000D            bsbw    outchar         ;output character
                F6      11      0010            brb     20$             ;
                        0012
52      08 AC   9A      0012    30$:    movzbl  size(ap),r2             ;maximum number of characters to read
                71      13      0016            beql    120$            ;if eql none
59      0C AC   D0      0018            movl    buf(ap),r9              ;set address of input buffer
                89      94      001C            clrb    (r9)+           ;initialize string count
                02 52   F5      001E            sobgtr  r2,40$          ;decrement and test character count
                53      11      0021            brb     110$            ;end of read
                        0023
05 10 AC        06      E0      0023    40$:    bbs     #6,option(ap),50$       ;if set, vt100 console terminal
                FFD5'   30      0028            bsbw    qvss$input      ;read character from qvss
                0A      11      002B            brb     60$             ;
```

```
                        002D   115
      50    20    DB    002D   116 50$:    mfpr    #pr$_rxcs,r0            ;receiver ready?
  F9  50    07    E1    0030   117         bbc     #7,r0,50$               ;if clr, receiver not ready
      50    21    DB    0034   118         mfpr    #pr$_rxdb,r0            ;read input character
58    50  80  8F    8B  0037   119 60$:    bicb3   #^x80,r0,r8             ;clear parity bit
58    58  7F  8F    91  003C   120         cmpb    #rubout,r8              ;rubout?
          11    12    0040   121         bneq    80$                     ;if neq no
      58    79    9A    0042   122         movzbl  -(r9),r8                ;get character to rubout
            CB    13    0045   123         beql    30$                     ;if eql none
   02 54    00    E2    0047   124         bbss    #v_rub,r4,70$           ;set start of rubout sequence
            40    10    004B   125         bsbb    outbslsh                ;output back slash
            44    10    004D   126 70$:    bsbb    outr8                   ;output rubbed out character
            52    D6    004F   127         incl    r2                      ;adjust remaining character count
            D0    11    0051   128         brb     40$                     ;
                        0053   129
   02 54    00    E5    0053   130 80$:    bbcc    #v_rub,r4,90$           ;terminate rubout sequence
            34    10    0057   131         bsbb    outbslsh                ;output backslash
      58    15    91    0059   132 90$:    cmpb    #control_u,r8           ;control u?
            A4    13    005C   133         beql    10$                     ;if eql yes
   03 58    06    E1    005E   134         bbc     #6,r8,100$              ;if clr, then graphic
      58    20    8A    0062   135         bicb    #32,r8                  ;convert to upper case
      50    0D    91    0065   136 100$:   cmpb    #cr,r0                  ;carriage return?
            0C    13    0068   137         beql    110$                    ;if eql yes
            52    D5    006A   138         tstl    r2                      ;any space left in buffer?
            B5    13    006C   139         beql    40$                     ;if eql no
            23    10    006E   140         bsbb    outr8                   ;echo character
      89    58    90    0070   141         movb    r8,(r9)+                ;buffer new character
        AD  52    F4    0073   142         sobgeq  r2,40$                  ;reduce space remaining (always loop)
                        0076   143
      58    0D    9A    0076   144 110$:   movzbl  #cr,r8                  ;set carriage return character
            1B    10    0079   145         bsbb    outchar                 ;
      50    0A    9A    007B   146         movzbl  #lf,r0                  ;yes send line feed also
            16    10    007E   147         bsbb    outchar                 ;output character in r0
   59     0C AC    C2    0080   148         subl    buf(ap),r9              ;compute character count + 1
0C BC 59    01    83    0084   149         subb3   #1,r9,@buf(ap)          ;set actual character count
      50    01    3C    0089   150 120$:   movzwl  #ss$_normal,r0          ;return normal completion status
                  04    008C   151         ret                             ;
                        008D   152
                        008D   153 outbslsh:                              ;output back slash
   50     5C 8F    9A    008D   154         movzbl  #^a%\%,r0               ;set character code
            03    11    0091   155         brb     outchar                 ;and output it
                        0093   156
      50    58    9A    0093   157 outr8:   movzbl  r8,r0                  ;get character to output
                        0096   158 outchar:                               ;output character in r0
03 10 AC    06    E0    0096   159         bbs     #6,option(ap),10$       ;if set, vt100 console terminal
          FF62'   31    009B   160         brw     qvss$output             ;
                        009E   161
      51    20    DB    009E   162 10$:    mfpr    #pr$_rxcs,r1            ;receiver ready?
1B    51    07    E1    00A1   163         bbc     #7,r1,30$               ;if clr, receiver not ready
      51    21    DB    00A5   164         mfpr    #pr$_rxdb,r1            ;read input character.
13    51    07    00    ED  00A8   165         cmpzv   #0,#7,r1,#control_s     ;control-s?
            11    12    00AD   166         bneq    30$                     ;if neq no
      51    20    DB    00AF   167 20$:    mfpr    #pr$_rxcs,r1            ;receiver ready?
  F9  51    07    E1    00B2   168         bbc     #7,r1,20$               ;if clr, receiver not ready
      51    21    DB    00B6   169         mfpr    #pr$_rxdb,r1            ;read input character
11    51    07    00    ED  00B9   170         cmpzv   #0,#7,r1,#control_q     ;is it a control-q?
            EF    12    00BE   171         bneq    20$                     ;no, wait for another character.
```

```
                51   22   DB  00C0   172 30$:    mfpr    #pr$_txcs,r1            ;transmitter done?
        F9      51   07   E1  00C3   173         bbc     #7,r1,30$               ;if clr, transmitter not done
                23   50   DA  00C7   174         mtpr    r0,#pr$_txdb            ;write output character
                          05  00CA   175         rsb                             ;return
                              00CB   176
                              00CB   177         .end
```

```
BOO$READPROMPT                    00000000 RG    02
BOOT_UV1_SWITCH                 = 00000001
BUF                             = 0000000C
CONTROL_Q                       = 00000011
CONTROL_S                       = 00000013
CONTROL_U                       = 00000015
CR                              = 0000000D
LF                              = 0000000A
OPTION                          = 00000010
OUTBSLSH                          0000008D R     02
OUTCHAR                           00000096 R     02
OUTR8                             00000093 R     02
PQ                              = 00000001   G
PR$_RXCS                        = 00000020
PR$_RXDB                        = 00000021
PR$_TXCS                        = 00000022
PR$_TXDB                        = 00000023
PROMPT                          = 00000004
QVSS$INPUT                        ********    X   02
QVSS$OUTPUT                       ********    X   02
RUBOUT                          = 0000007F
SIZE                            = 00000008
SS$_NORMAL                      = 00000001
V_RUB                           = 00000000
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
$CONIO	000000CB (203.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	9	00:00:00.07	00:00:00.43
Command processing	84	00:00:00.66	00:00:01.50
Pass 1	173	00:00:04.54	00:00:05.87
Symbol table sort	0	00:00:00.74	00:00:00.75
Pass 2	37	00:00:00.94	00:00:01.28
Symbol table output	4	00:00:00.04	00:00:00.04
Psect synopsis output	1	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	308	00:00:07.03	00:00:09.91

The working set limit was 900 pages.
25745 bytes (51 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 506 non-local and 15 local symbols.
179 source lines were read in Pass 1, producing 16 object records in Pass 2.
9 pages of virtual memory were used to define 8 macros.

```
                          +---------------------------+
                          ! Macro library statistics !
                          +---------------------------+

Macro library name                                 Macros defined
------------------                                 --------------
DISK$STARWORK03:[GAMACHE.UV1ROM.VMS]LIBUV1.ML                   0
DISK$STARWORK03:[GAMACHE.UV1ROM.OBJ]VMB.MLB;3                   0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2                               5
TOTALS (all libraries)                                          5

553 GETS were required to define 5 macros.

There were no errors, warnings or information messages.

MAC/LIS=LIS$:CONIO/OBJ=OBJ$:CONIO VMS$:BOOUV1SWT+VMB$:CONIO+OBJ$:VMB/LIB+VMS$:LIBUV1/LIB
```

LUV1ROM

PQBTDRIVR
LIS

QVSS
LIS

SEARCHMSG
LIS

VMBUVAX1
MAP

SETUSER
LIS

VMBUVAX1
LIS

BOOTDRUV1
LIS

FILERDUV1
LIS

BOOTIOUV1
LIS

CONIO
LIS

NETBOOT
LIS